REMARKS

Initially, Applicants note, that in an attempt to expedite prosecution of this application, a request for an interview after final rejection was made to clarify the distinctions between the claimed invention and the references applied by the Examiner. The Examiner denied the interview request.

In the final Office Action, the Examiner rejected claims 1, 2, 4-7, 10-13, and 20-25 under 35 U.S.C. § 102(b) as anticipated by Mattis et al. (U.S. Patent No. 6,209,003); rejected claim 3 under 35 U.S.C. § 103(a) as unpatentable over Mattis et al. in view of Manley et al. (U.S. Patent Application Publication No. 2003/0182330); and rejected claims 8, 9, and 14-19 under 35 U.S.C. § 103(a) as unpatentable over Mattis et al. in view of Hisgen et al. ("New-Value Logging in the Echo Replicated File System," June 23, 1993).

Applicants respectfully traverse the Examiner's rejections under 35 U.S.C. §§ 102 and 103. Claims 1-25 remain pending.

REJECTION UNDER 35 U.S.C. § 102 BASED ON MATTIS ET AL.

In paragraph 6 of the final Office Action, the Examiner rejected claims 1, 2, 4-7, 10-13, and 20-25 under 35 U.S.C. § 102(b) as allegedly anticipated by Mattis et al. Applicants respectfully traverse the rejection.

A proper rejection under 35 U.S.C. § 102 requires that a single reference teach every aspect of the claimed invention. Any feature not directly taught must be inherently present. In other words, the identical invention must be shown in as complete detail as contained in the claim. See M.P.E.P. § 2131. Mattis et al. does not disclose or suggest the combination of features recited in claims 1, 2, 4-7, 10-13, and 20-25.

For example, independent claim 1 is directed to a method for deleting one or more of a plurality of files, where the files include one or more chunks stored by a plurality of servers. The method comprises identifying a file to be deleted; renaming the identified file; permanently deleting the renamed file a predetermined amount of time after renaming the identified file as part of a garbage collection process; receiving, from the servers, information concerning chunks stored by the servers; and identifying, to one of the servers, one of the chunks that corresponds to the permanently deleted file.

Mattis et al. does not disclose or suggest the combination of features recited in claim 1. For example, Mattis et al. does not disclose or suggest renaming a file that is identified to be deleted. The Examiner alleged that Mattis et al. discloses this feature and cited column 3, lines 16-19, of Mattis et al. for support (final Office Action, page 3). Applicants respectfully disagree.

At column 3, lines 13-19, Mattis et al. discloses:

> Also, file systems, being designed for user data file management, include facilities irrelevant to cache object stores, and indeed counter-productive to this application. Examples include: support for random access and selective modification, file permissions, support for moving files, support for renaming files, and support for appending to files over time.

This section of Mattis et al. discloses that file systems, which are a prior art approach to structure cache object stores, include support for renaming files. This portion specifically states that renaming files is "irrelevant to cache object stores" and "counter-productive to [the Mattis et al.] application" (col. 3, lines 13-19). In other words, this section of Mattis et al. specifically teaches away from renaming files. Instead, Mattis et al. discloses that if a fragment is to be deleted, then it is deleted by marking it as deleted and overwriting the data in the fragment (col. 23, lines 15-17). Nowhere does Mattis et al. disclose or suggest renaming a file that is identified to be

deleted, as required by claim 1.

Nevertheless, this section of Mattis et al. discloses a prior art approach that supports renaming files. Contrary to the Examiner's allegation, however, this prior art approach described by Mattis et al. does not disclose or remotely suggest renaming a file that is identified to be deleted, as required by claim 1.

Further, even if, for the sake of argument, this prior art approach described by Mattis et al. could be equated to renaming a file that is identified to be deleted (a point that Applicants submit is unreasonable), the Examiner has identified a prior art approach as allegedly equivalent to this feature of claim 1 but relied upon the system of Mattis et al. for other features of claim 1. As noted above, the Mattis et al. system does not include this prior art approach and specifically teaches away from including this prior art approach, and the Examiner has not provided the requisite motivation for modifying the Mattis et al. system to include this prior art approach. Accordingly, the Examiner has not established a proper rejection under 35 U.S.C. § 102 with regard to claim 1.

The Examiner also alleged that Mattis et al. discloses:

> a method for garbage collection, which means it is a method for detecting and freeing (i.e. deletion) of unwanted memory. Therefore, since the sole of the invention is to identify space that needs to be deleted, it is common practice that the renaming of files would be assigned to the files, which were identified as being deleted. To further elaborate, Mattis discloses at column 23, lines 15-23, wherein if a fragment is to be deleted, the fragment is marked as deleted, which means that the fragment is an altered fragment that has now been renamed because of the deletion status. As a result, Mattis discloses renaming a file that is identified to be deleted.

(final Office Action, page 10). The Examiner's allegation does not find support in the disclosure of Mattis et al. The Examiner alleged that "it is common practice that the renaming of files would be assigned to the files" (emphasis added) (final Office Action, page 10), but the

Examiner has not identified any portion of the disclosure of Mattis et al. or any other evidence that supports the Examiner's allegation.

At column 23, lines 15-23, Mattis et al. discloses:

> If the fragment is to be deleted, then in step 812 it is deleted from the arena by marking it as deleted and overwriting the data in the fragment. When an object 52 is stored in multiple fragments, and the garbage collection process determines that one of the fragments is to be deleted, then the process deletes all fragments associated with the object. This may involve following a chain of fragments, of the type shown in FIG. 5, to another arena or even another pool.

In this section, Mattis et al. discloses that if a fragment is to be deleted, it is deleted by marking it as deleted and overwriting the data in the fragment. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest renaming a file that is identified to be deleted, as required by claim 1.

Mattis et al. also does not disclose or suggest permanently deleting the renamed file a predetermined amount of time after renaming the identified file as part of a garbage collection process. The Examiner alleged that Mattis et al. discloses this feature and cited column 16, lines 48-52; column 21, lines 52-55; and column 22, lines 43-47, of Mattis et al. for support (final Office Action, page 3). Applicants respectfully disagree.

At column 16, lines 48-52, Mattis et al. discloses:

> For example, the Open Directory is useful in safeguarding against overwriting or deleting an object that is currently being read. The Open Directory also buffers changes to the Directory Table 110 before they are given permanent effect in the Directory Table 110.

In this section, Mattis et al. discloses that the Open Directory buffers changes to the Directory Table before they are given permanent effect in order to safeguard against overwriting or deleting an object that is currently being read. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest permanently deleting a renamed file a predetermined amount of

time after renaming a file that is identified to be deleted as part of a garbage collection process, as required by claim 1.

The Examiner alleged that this section of Mattis et al. means that the act of deleting is buffered to make sure that it is the correct action that needs to be taken and, if so, the deletion is made permanent (final Office Action, page 11). Applicants respectfully submit that the Examiner is misinterpreting this section of Mattis et al. This section states that changes to the Directory Table are buffered to safeguard against overwriting or deleting an object that is currently being read. This section does not, as alleged by the Examiner, state that a deletion is buffered to make sure that it is the correct action that needs to be taken. Therefore, the Examiner's interpretation of this section of Mattis et al. is flawed.

At column 21, lines 52-55, Mattis et al. discloses:

> Preferably, the garbage collection method is implemented as an independent process that runs in parallel with other processes that relate to the cache. This enables the garbage collection method to periodically clean up cache storage areas without interrupting or affecting the operation of the cache.

In this section, Mattis et al. discloses that a garbage collection method periodically cleans up the cache storage areas. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest permanently deleting a renamed file a predetermined amount of time after renaming a file that is identified to be deleted as part of a garbage collection process, as required by claim 1.

At column 22, lines 39-47, Mattis et al. discloses:

> In step 806, one of the fragments within the selected arena is selected for garbage collection. In determining which fragment or fragments to select, the cache 80 takes into account several selection factors, as indicated by block 807. In the preferred embodiment, the factors include: the time of the last access to the fragment; the number of hits that have occurred to an object that has data in the fragment; the time required to download data from the fragment to a client; and the size of the object of which the fragment is a part.

In this section, Mattis et al. discloses factors that are taken into consideration when determining which fragment to select for garbage collection. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest permanently deleting a renamed file a predetermined amount of time after renaming a file that is identified to be deleted as part of a garbage collection process, as required by claim 1.

The Examiner alleged that the factors that the Mattis et al. system considers when determining which fragment to select for garbage collection "demonstrate a predetermined amount of time after renaming" (final Office Action, page 11). Applicants submit that the Examiner's conclusion finds no support in the disclosure of Mattis et al. and falls short of establishing a proper rejection under 35 U.S.C. § 102.

The Examiner also alleged that

> [t]he limitation of "permanently deleting a file" has not been entirely disclosed within [Applicants'] specification, since even though a user may mark a file as deleted and proceed to further remove the file, it is known that the "deleted/removed file" is truly not permanently deleted, because the information is still within the computer system and if needed, could possibly be retrieved

(emphasis in original) (final Office Action, pages 10-11). Applicants submit that Applicants' original specification provides full support for the claimed features. For example, support for the claim feature of "permanently deleting the renamed file . . . " can be found in the specification at paragraph 0072. Therefore, the Examiner's allegation that this feature is not disclosed in the specification is without merit.

The Examiner further cited column 32, lines 29-37, of Mattis et al. for allegedly disclosing that a block is set with a deletion flag indicating that the block will ultimately be deleted and eventually, the block is removed from the directory, thereby permanently deleting

the file (final Office Action, page 11). At column 32, lines 29-37, Mattis et al. discloses:

> To accomplish removal of a block found in the cache, however, in step 960 the process sets the deletion flag, and checks the block in with the deletion flag set. As described herein in connection with the check-in process (steps 938 and 944 of FIG. 9B), when the deletion flag is set, the block will be marked as deleted. Thereafter, the block is eventually removed from the Directory Index when the changes reflected in the Open Directory are synchronized to the Directory Index.

In this section, Mattis et al. discloses that to remove a block, a deletion flag is set, the block is marked as deleted, and the block is eventually removed from the Directory Index. Contrary to the Examiner's allegation, nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest permanently deleting a renamed file a predetermined amount of time after renaming a file that is identified to be deleted as part of a garbage collection process, as required by claim 1.

The Examiner further cited column 22, lines 14-23, of Mattis et al. for allegedly disclosing

> high and low water marks need to be at a certain value, and if the water marks are not at those values the garbage collection is carried out at a time before the capacity is exceeded. This example not only discloses a predetermined amount of time, but it also discloses the fact this is all part of a garbage collection process. As a result, Mattis discloses permanently deleting the renamed file a predetermined amount of time after renaming the identified file as part of a garbage collection process.

(final Office Action, page 11). Applicants submit that the Examiner's conclusion finds no support in the disclosure of Mattis et al. and results solely from the use of impermissible hindsight reasoning.

At column 22, lines 14-23, Mattis et al. discloses:

> When the amount of active storage in a particular pool becomes greater than the "high water mark" value, garbage collection is initiated and carried out repeatedly until the amount of active storage in the pool falls below the "low water mark" value. The "low water mark" value is selected to be greater than zero, and the "high water mark" value is chosen to be approximately 20% less than the total storage capacity of the pool. In this way, garbage collection is carried out at a time before the pool overflows or the capacity of the storage device 90a is exceeded.

In this section, Mattis et al. discloses that garbage collection is initiated when the amount of active storage in the pool becomes greater than a high water mark value and continues until the amount of active storage in the pool falls below a low water mark value. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest permanently deleting a renamed file a predetermined amount of time after renaming a file that is identified to be deleted as part of a garbage collection process, as required by claim 1.

Claim 1 recites additional features that are neither disclosed nor suggested by Mattis et al. For example, Mattis et al. does not disclose or suggest identifying, to one of the servers, one of the chunks that corresponds to the permanently deleted file, as further recited in claim 1. The Examiner alleged that Mattis et al. discloses this feature and cited column 33, lines 32-40, of Mattis et al. for support (final Office Action, page 3). Applicants respectfully disagree.

At column 33, lines 32-40, Mattis et al. discloses:

> If no match of the key is found in the search, then in step 848 the process returns an error message to the calling program or process, indicating that the requested object does not exist in the cache. Although the specific response to such a message is determined by the calling program or process, in the World Wide Web context, generally the proxy 30 contacts the server 40 that stores the object using an HTTP request, and obtains a copy of the requested object.

In this section, Mattis et al. discloses that a search is performed for a key to determine whether a requested object exists in the cache and if no match of the key is found, the proxy contacts the server to obtain a copy of the requested object. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest identifying, to one of the servers, one of the chunks that corresponds to a file that was permanently deleted a predetermined amount of time after renaming the file, as required by claim 1.

The Examiner alleged that from this section of Mattis et al.

> [t]he requested object is known as being permanently deleted because initially the search receives an error message stating the object does not exist. If the object does not exist it is obvious that the object was permanently deleted. Therefore, Mattis does disclose identifying, to one of the servers, one of the chunks that corresponds to the permanently deleted file.

(final Office Action, page 12). Applicants respectfully submit that the Examiner does not understand the basic operation of a cache. When a cache miss occurs (i.e., when a request for an object from the cache is made and the object is not present in the cache), this does not necessarily mean that object was once present in the cache and subsequently permanently deleted. Instead, it simply means that the object is currently not stored by the cache. For example, it is possible that the object was never stored by the cache. Thus, the Examiner's allegation is flawed and the Examiner's conclusion, which is based on this erroneous assumption, is equally flawed and finds no support in the disclosure of Mattis et al.

For at least these reasons, Applicants submit that claim 1 is not anticipated by Mattis et al. Claims 2, 4-7, 10, and 11 depend from claim 1 and are, therefore, not anticipated by Mattis et al. for at least the reasons given with regard to claim 1. Claims 2, 4-7, 10, and 11 are also not anticipated by Mattis et al. for reasons of their own.

For example, claim 4 recites that the predetermined amount of time is a user-configurable amount of time. As explained above with regard to claim 1, Mattis et al. is completely silent with regard to permanently deleting a renamed file a predetermined amount of time after renaming the file. Therefore, Mattis et al. cannot disclose or suggest that the predetermined amount of time is a user-configurable amount of time, as required by claim 4. In other words, Mattis et al. is completely silent with regard to permanently deleting a renamed file a user-

configurable amount of time after renaming the file, as required by claim 4.

The Examiner alleged that Mattis et al. discloses this feature and cited column 22, lines 14-23, of Mattis et al. for support (final Office Action, page 3). Applicants respectfully disagree.

Column 22, lines 14-23, of Mattis et al. is reproduced above. In this section, Mattis et al. discloses that garbage collection is initiated when the amount of active storage in a pool becomes greater than a high water mark. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest permanently deleting a renamed file a user-configurable amount of time after renaming the file, as required by claim 4.

The Examiner alleged that the high and low water marks in this section of Mattis et al.

> were assigned within the application during the programming stages, which is predetermined and done by the user (i.e. programmer). Therefore, Mattis does disclose the predetermined amount of time is a user-configurable amount of time.

(final Office Action, pages 12-13). The Examiner's allegation finds no basis in fact. As explained above with regard to claim 1, Mattis et al. does not disclose or suggest permanently deleting a renamed file a predetermined amount of time after renaming the file. Therefore, even assuming, for the sake of argument, that a programmer of an application in Mattis et al. can be considered a user (a point that Applicants do not concede), Mattis et al. does not disclose or remotely suggest permanently deleting a renamed file a user-configurable amount of time after renaming the file, as required by claim 4. The Examiner's allegations to the contrary find no support in the disclosure of Mattis et al. and result solely from the use of impermissible hindsight reasoning.

For at least these additional reasons, Applicants submit that claim 4 is not anticipated by Mattis et al. Claim 5 depends from claim 4 and is, therefore, also not anticipated by Mattis et al.

for at least the reasons given with regard to claim 4.

Claim 10 recites maintaining versions of the chunks; identifying a stale chunk based on the versions of the chunks; and deleting the stale chunk. Mattis et al. does not disclose or suggest the combination of features recited in claim 10. For example, Mattis et al. does not disclose or suggest identifying a stale chunk based on the versions of the chunks.

The Examiner alleged that Mattis et al. discloses identifying a stale chunk based on the versions of the chunks and cited column 26, lines 15-21, of Mattis et al. for support (final Office Action, page 4). Applicants disagree.

At column 26, lines 15-22, Mattis et al. discloses:

> In block 1216, the cache updates an expiration date value stored in association with the information object to reflect the current date or time. By updating the expiration date, the cache ensures that the garbage collection process will not delete the object, because after the update it is not considered old. In this way, an old object is refreshed in the cache without retrieving the object from its origin, writing it in the cache, and deleting a stale copy of the object.

In this section, Mattis et al. discloses that the cache updates an expiration date value stored in association with an information object. Applicants submit that nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest identifying a stale chunk based on the versions of the chunks, as required by claim 10.

Even assuming, for the sake of argument, that this section of Mattis et al. can reasonably be understood to disclose identifying a stale chunk (a point that Applicants do not concede), Applicants submit that the Examiner's rejection is flawed. The Examiner alleged that the version number of the program or process that created an arena in a pool (col. 17, lines 42-46) is allegedly equivalent to a version of a chunk (final Office Action, page 4). With this interpretation in mind, nowhere in column 26, lines 15-21, does Mattis et al. disclose or remotely

suggest identifying a stale chunk based on a version number of a program or process that created an arena in a pool. Therefore, column 26, lines 15-21, of Mattis et al. cannot disclose or suggest identifying a stale chunk based on the versions of the chunks, as required by claim 10.

For at least these additional reasons, Applicants submit that claim 10 is not anticipated by Mattis et al. Claim 11 depends from claim 10 and is, therefore, also not anticipated by Mattis et al. for at least the reasons given with regard to claim 10.

Independent claims 12 and 13 recite features similar to, but possibly different in scope from, features recited in claim 1. Claims 12 and 13 are, therefore, not anticipated by Mattis et al. for at least reasons similar to reasons given with regard to claim 1.

Independent claim 20 is directed to a method for deleting stale replicas of chunks, where the replicas are stored by a plurality of servers. The method comprises associating version information with replicas of chunks; identifying stale replicas based on the associated version information; deleting the stale replicas; receiving, from the servers, information concerning replicas stored by the servers; and identifying, to one of the servers, one of the replicas that corresponds to one of the deleted stale replicas.

Mattis et al. does not disclose or suggest the combination of features recited in claim 20. For example, Mattis et al. does not disclose or suggest identifying stale replicas based on the associated version information.

The Examiner alleged that Mattis et al. discloses identifying stale replicas based on the associated version information and identified column 26, lines 15-21, of Mattis et al. for support (final Office Action, page 5). Applicants respectfully disagree.

Column 26, lines 15-22, of Mattis et al. is reproduced above. In this section, Mattis et al.

discloses that the cache uses an expiration date value to determine whether to delete an object from the cache. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest, for example, identifying stale replicas based on the associated version information, as required by claim 20.

Even assuming, for the sake of argument, that this section of Mattis et al. can reasonably be understood to disclose identifying stale replicas (a point that Applicants do not concede), Applicants submit that the Examiner's rejection is flawed. The Examiner alleged that the different versions of an object (e.g., versions in English, French, and Japanese languages) (col. 14, lines 29-37) is allegedly equivalent to the generated version information for replicas of chunks (final Office Action, page 5). With this interpretation in mind, nowhere in column 26, lines 15-21, does Mattis et al. disclose or remotely suggest identifying stale replicas based on this information regarding different versions of an object. Therefore, column 26, lines 15-21, of Mattis et al. cannot disclose or suggest identifying stale replicas based on the associated version information, as required by claim 20.

Mattis et al. also does not disclose or suggest identifying, to one of the servers, one of the replicas stored by the server that corresponds to one of the deleted stale replicas.

The Examiner alleged that Mattis et al. discloses identifying, to one of the servers, one of the replicas that corresponds to one of the deleted stale replicas and identified column 26, lines 15-21, of Mattis et al. for support (final Office Action, page 5). Applicants respectfully disagree.

Column 26, lines 15-21, of Mattis et al. is reproduced above. In this section, Mattis et al. discloses that the cache uses an expiration date value to determine whether to delete an object from the cache. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest, for

example, identifying, to one of the servers, one of the replicas stored by the server that corresponds to one of the deleted stale replicas, as required by claim 20.

If the Examiner persists with the rejection of this feature of claim 20, Applicants respectfully request that the Examiner provide a reasonable explanation of how column 26, lines 15-21, of Mattis et al. can correspond to identifying, to one of the servers, one of the replicas stored by the server that corresponds to one of the deleted stale replicas, as required by claim 20.

For at least these reasons, Applicants submit that claim 20 is not anticipated by Mattis et al. Claims 21-23 depend from claim 20 and are, therefore, not anticipated by Mattis et al. for at least the reasons given with regard to claim 20. Claims 21-23 are also not anticipated by Mattis et al. for reasons of their own.

For example, claim 21 recites that the version information for one of the replicas is updated each time a lease is granted for the one of the replicas. Mattis et al. does not disclose or suggest this feature.

The Examiner alleged that Mattis et al. discloses version information for one of the replicas that is updated each time a lease is granted for the one of the replicas and cited column 26, lines 8-15, of Mattis et al. for support (final Office Action, page 6). Applicants disagree.

At column 26, lines 8-15, Mattis et al. discloses:

> If the Read Counter value is high, then the information object has been loaded recently. In that case, in block 1210 the cache sends a positive response message to the requesting process. Otherwise, as indicated in block 1212, the information object has not been loaded recently. Accordingly, as shown in block 1214, the cache sends a negative responsive message to the calling program or process.

In this section, Mattis et al. discloses if the read counter value is high, the cache sends a positive response message to a requesting process; otherwise, the cache sends a negative response

message to the calling program or process. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest version information for one of the replicas that is updated each time a lease is granted for the one of the replicas, as required by claim 21.

Applicants also note that the Examiner alleged that the different versions of an object (e.g., versions in English, French, and Japanese languages) (col. 14, lines 29-37) is allegedly equivalent to the generated version information for replicas of chunks (final Office Action, page 5). With this interpretation in mind, nowhere in column 26, lines 8-15, does Mattis et al. disclose or remotely suggest that the different versions of an object are updated each time a lease is granted for the object. Therefore, column 26, lines 8-15, of Mattis et al. cannot disclose or suggest version information for one of the replicas that is updated each time a lease is granted for the one of the replicas, as required by claim 21.

For at least these additional reasons, Applicants submit that claim 21 is not anticipated by Mattis et al.

Independent claims 24 and 25 recite features similar to, but possibly different in scope from, features recited in claim 20. Claims 24 and 25 are, therefore, not anticipated by Mattis et al. for at least reasons similar to reasons given with regard to claim 20.

Accordingly, Applicants respectfully request reconsideration and withdrawal of the rejection of claims 1, 2, 4-7, 10-13, and 20-25 under 35 U.S.C. § 102 based on Mattis et al.

REJECTION UNDER 35 U.S.C. § 103 BASED ON MATTIS ET AL. AND MANLEY ET AL.

In paragraph 8 of the final Office Action, the Examiner rejected claim 3 under 35 U.S.C. § 103(a) as allegedly unpatentable over Mattis et al. in view of Manley et al. Applicants respectfully traverse the rejection.

Initially, claim 3 depends on claim 1. The disclosure of Manley et al. does not cure the deficiencies in the disclosure of Mattis et al. identified above with regard to claim 3. Therefore, claim 3 is patentable over Mattis et al. and Manley et al., whether taken alone or in any reasonable combination, for at least the reasons given with regard to claim 1. Claim 3 is also patentable over Mattis et al. and Manley et al. for reasons of its own.

Claim 3 recites receiving an un-deletion instruction regarding the file, and restoring an original name to the file without permanently deleting the renamed file. Neither Mattis et al. nor Manley et al. discloses or suggests the combination of features recited in claim 3. For example, neither Mattis et al. nor Manley et al. discloses or suggests restoring an original name to a renamed file without permanently deleting the renamed file.

The Examiner admitted that Mattis et al. does not disclose this feature, but alleged that Manley et al. discloses the feature and cited paragraph 0067 of Manley et al. for support (final Office Action, page 7). Applicants respectfully disagree.

At paragraph 0067, Manley et al. discloses:

> FIG. 7 shows an exemplary inode file system structure 700 after a file data block has been modified. In this illustrative example, file data which is stored at disk block 520C is modified. The exemplary WAFL file system writes the modified contents to disk block 520C', which is a new location on disk. Because of this new location, the inode file data which is stored at disk block (515) is rewritten so that it points to block 520C'. This modification causes WAFL to allocate a new disk block (715) for the updated version of the data at 515. Similarly, the inode file indirect block 510 is rewritten to block 710 and direct block 512 is rewritten to block 712, to point to the newly revised inode 715. Thus, after a file data block has been modified the snapshot inode 605 contains a pointer to the original inode file system indirect block 510 which, in turn, contains a link to the inode 515. This inode 515 contains pointers to the original file data blocks 520A, 520B and 520C. However, the newly written inode 715 includes pointers to unmodified file data blocks 520A and 520B. The inode 715 also contains a pointer to the modified file data block 520C' representing the new arrangement of the active file system. A new file system root inode 705 is established representing the new structure 700. Note that metadata in any snapshotted blocks (e.g. blocks 510, 515 and 520C) protects these blocks from being recycled or overwritten until they are released from all snapshots. Thus, while

> the active file system root 705 points to new blocks 710, 712, 715 and 520C', the old blocks 510, 515 and 520C are retained until the snapshot is fully released.

In this section, Manley et al. discloses that after a file data block has been modified, the snapshot inode contains a pointer to the original inode file system indirect block which contains a link to the inode. Nowhere in this section, or elsewhere, does Manley et al. disclose or suggest restoring an original name to a renamed file without permanently deleting the renamed file, as required by claim 3.

The Examiner also alleged that because "there is a pointer to the original file then the original name must be re-assigned" (final Office Action, page 14). Applicants submit that the Examiner is not addressing the feature of claim 3. Claim 3 does not recite a point to an original file or reassigning an original name, but instead recites restoring an original name to a renamed file without permanently deleting the renamed file. The Examiner's allegation falls short of establishing a prima facie case of obviousness with regard to claim 3.

The Examiner also cited paragraph 0118 of Manley et al. for allegedly disclosing restoring an original name to a renamed file without permanently deleting the renamed file (final Office Action, page 14). Applicants disagree.

At paragraph 0118, Manley et al. discloses:

> However, in the illustrative embodiment, if the source inodes received at the destination refer to inodes in the inode map 1400, then the directory stage creates (on the current built-up snapshot directory 1330) a file entry having the desired file name. This name can be exactly the name derived from the source. A hard link 1332 (i.e. a Unix-based link enables multiple names to be assigned to a discrete file) is created between that file on the snapshot directory 1330 and the entry in the purgatory directory. By so linking the entry, it is now pointed to by both the purgatory directory and the file on the snapshot directory itself. When the purgatory directory root is eventually deleted (thereby killing off purgatory) at the end of the data stream transfer, the hard link will remain to the entry, ensuring that the specific entry in the purgatory directory will not be deleted or recycled (given that the entry's link count is still greater than zero) and a path to the data from the file on the new directory is maintained. Every purgatory entry that eventually becomes

> associated with a file in the newly built tree will be similarly hard linked, and thereby survive deletion of the purgatory directory. Conversely, purgatory entries that are not relinked will not survive, and are effectively deleted permanently when purgatory is deleted.

In this section, Manley et al. discloses a purgatory entry that is hard linked will survive deletion of the purgatory directory and that a purgatory entry that is not relinked will not survive and will be deleted permanently. Nowhere in this section, or elsewhere, does Manley et al. disclose or suggest restoring an original name to a renamed file without permanently deleting the renamed file, as required by claim 3.

The Examiner alleged that this section of Manley et al.

> means that the file has been marked to be deleted was given a hard link to the original file, represented by the specific entry, so the path to the data remains. Once the file is un-deleted, the file survives deletion and is given its original name because of the original linkage. As a result, Mattis in view of Manley, disclose restoring an original name to a renamed file without permanently deleting the renamed file.

(final Office Action, page 14). As explained above, nowhere in paragraph 0118, or elsewhere, does Manley et al. disclose or suggest restoring an original name to a renamed file without permanently deleting the renamed file, as required by claim 3. The Examiner's allegations to the contrary find no support in the disclosure of Manley et al. and result solely from the use of impermissible hindsight reasoning.

For at least these reasons, Applicants submit that claim 3 is patentable over Mattis et al. and Manley et al., whether taken alone or in any reasonable combination.

Accordingly, Applicants respectfully request reconsideration and withdrawal of the rejection of claim 3 under 35 U.S.C. § 103 based on Mattis et al. and Manley et al.

REJECTION UNDER 35 U.S.C. § 103 BASED ON MATTIS ET AL. AND HISGEN ET AL.

In paragraph 9 of the final Office Action, the Examiner rejected claims 8, 9, and 14-19

under 35 U.S.C. § 103(a) as allegedly unpatentable over Mattis et al. in view of Hisgen et al. Applicants respectfully traverse the rejection.

Claims 8 and 9 depend from claim 1. Without acquiescing in the Examiner's rejection, Applicants submit that the disclosure of Hisgen et al. does not cure the deficiencies in the disclosure of Mattis et al. identified above with regard to claim 1. Therefore, claims 8 and 9 are patentable over Mattis et al. and Hisgen et al., whether taken alone or in any reasonable combination, for at least the reasons given with regard to claim 1.

Independent claim 14 is directed to a method for deleting orphaned chunks of a plurality of chunks stored by a plurality of servers. The method comprises providing a mapping of file names to chunks; identifying chunks, as orphaned chunks, that are not reachable from any of the file names; deleting the orphaned chunks; receiving, from the servers, information concerning chunks stored by the servers; and identifying, to one of the servers, one of the chunks that corresponds to one of the deleted orphaned chunks.

Neither Mattis et al. nor Hisgen et al., whether taken alone or in any reasonable combination, discloses or suggests the combination of features recited in claim 14. For example, neither Mattis et al. nor Hisgen et al. discloses or suggests identifying, to one of the servers, one of the chunks that corresponds to one of the deleted orphaned chunks.

The Examiner admitted that Mattis et al. does not disclose this feature, but alleged that Hisgen et al. discloses identifying, to the servers, ones of the chunks that are orphaned chunks and cited page 24, paragraph 3, lines 4-5, of Hisgen et al. for support (final Office Action, page 9). Applicants respectfully disagree.

At page 24, paragraph 3, Hisgen et al. discloses:

> Orphan list. An orphan file is a file that is no longer in the name space, in that no entry in any directory points to it, but it is still open on some client machine and therefore must still be kept in existence. (We use the term "orphan" because an orphan file has no parent directory.) The orphan list lists all orphan files. When the Echo distributed client-server caching algorithm [21] reveals that nobody has an orphan file open, EchoBox can remove the file from its orphan list and delete the file, returning its pages to the disk space allocator and removing it from the fid map.

In this section, Hisgen et al. discloses that when an orphan file is not open, the file can be removed from the orphan list and deleted. Nowhere in this section, or elsewhere, does Hisgen et al. disclose or suggest identifying, to one of the servers from which information concerning chunks stored by the servers was received, one of the chunks that corresponds to one of the deleted orphaned chunks, as required by claim 14.

For at least these reasons, Applicants submit that claim 14 is patentable over Mattis et al. and Hisgen et al., whether taken alone or in any reasonable combination. Claims 15-17 depend from claim 14 and are, therefore, patentable over Mattis et al. and Hisgen et al. for at least the reasons given with regard to claim 14.

Independent claims 18 and 19 recite features similar to, but possibly different in scope from, features recited in claim 14. Claims 18 and 19 are, therefore, patentable over Mattis et al. and Hisgen et al., whether taken alone or in any reasonable combination, for at least reasons similar to reasons given with regard to claim 14.

Accordingly, Applicants respectfully request reconsideration and withdrawal of the rejection of claims 8, 9, and 14-19 under 35 U.S.C. § 103 based on Mattis et al. and Hisgen et al.

CONCLUSION

In view of the foregoing remarks, Applicants respectfully request the Examiner's reconsideration of the application and the timely allowance of pending claims 1-25.

As Applicants' remarks with respect to the Examiner's rejections overcome the rejections, Applicants' silence as to certain assertions by the Examiner in the Office Action or certain requirements that may be applicable to such rejections (e.g., whether a reference constitutes prior art, motivation to combine references, etc.) is not a concession by Applicants that such assertions are accurate or that such requirements have been met, and Applicants reserve the right to dispute these assertions/requirements in the future.

If the Examiner does not believe that all pending claims are now in condition for allowance, the Examiner is urged to contact the undersigned to expedite prosecution of this application.

To the extent necessary, a petition for an extension of time under 37 C.F.R. § 1.136 is hereby made. Please charge any shortage in fees due in connection with the filing of this paper, including extension of time fees, to Deposit Account No. 50-1070 and please credit any excess fees to such deposit account.

Respectfully submitted,

HARRITY SNYDER, LLP

By: /Paul A. Harrity/
Paul A. Harrity
Reg. No. 39,574

Date: August 14, 2006
11350 Random Hills Road
Suite 600
Fairfax, Virginia 22030
(571) 432-0800